

## **Compte rendu TP3 Java**

### **Sommaire Automatique**

Conversion du script BP1 -> exercice 4.....	2
Conversion script BP2->Question3 JAVA.....	5
Réponse au Quizz du TP.....	7
Exercice 1. Identifiants de classes.....	8
Exercice 2. Identifiants de méthodes.....	9
Exercice 3. ParoleChanson.java.....	10
Exercice 4. Modèle illustré.....	11
Exercice 5. Modèle illustré.....	12
Exercice 6:.....	13
Debug 1 JAVA.....	13
Debug 2 JAVA.....	14
Debug 3 JAVA.....	15
Debug 4 JAVA.....	16
Cas pratique.....	17

# Conversion du script BP1 -> exercice 4

```
Algorithme Moyenne
Début
    Déclarer notes[1..4] en réel
    Déclarer somme, moyenne en réel
    somme ← 0

    Pour i ← 1 à 4 faire
        Écrire "Saisir la note n°", i, " :"
        Lire notes[i]
        somme ← somme + notes[i]
    Fin Pour

    moyenne ← somme / 4
    Écrire "La moyenne est : ", moyenne
Fin
```

```
1  import java.util.Scanner;
2
3  public class MoyenneNotes {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6          double[] notes = new double[4];
7          double somme = 0;
8
9          for (int i = 0; i < 4; i++) {
10             System.out.print("Saisir la note n°" + (i + 1) + " : ");
11             notes[i] = scanner.nextDouble();
12             somme += notes[i];
13         }
14
15         double moyenne = somme / 4;
16         System.out.println("La moyenne est : " + moyenne);
17         scanner.close();
18     }
19 }
20
```

J'ai commencé par déclarer un tableau nommé `notes` de type `double` avec une taille de 4, destiné à stocker les quatre notes saisies par l'utilisateur. En Java, les indices des tableaux commencent à 0, ce qui signifie que les cases vont de 0 à 3.

La déclaration correspondante est la suivante :

```
double[] notes = new double[4];
```

Ensuite, j'ai déclaré deux variables de type `double` : `somme`, qui servira à accumuler la somme des notes, et `moyenne`, qui contiendra le résultat final du calcul. J'ai initialisé `somme` à 0 afin de pouvoir y additionner les notes au fur et à mesure de leur saisie.

Voici les lignes de code utilisées :

```
double somme = 0;  
double moyenne;
```

Pour permettre la saisie des notes par l'utilisateur, j'ai instancié un objet de la classe `Scanner`, qui facilite la lecture des données entrées au clavier.

La ligne correspondante est :

```
Scanner sc = new Scanner(System.in);
```

J'ai ensuite mis en place une boucle `for` allant de 0 à 3 (soit 4 itérations) afin de demander à l'utilisateur de saisir chaque note, une par une. À chaque itération, un message s'affiche pour indiquer le numéro de la note à saisir, en ajoutant 1 à l'indice de la boucle `i` pour que la numérotation commence à 1 et soit plus compréhensible pour l'utilisateur.

Ce bloc est écrit ainsi :

```
for (int i = 0; i < 4; i++) {  
    System.out.print("Saisir la note n°" + (i + 1) + " : ");
```

La note saisie est récupérée avec la méthode `nextDouble()` de l'objet `Scanner` et stockée dans la case `i` du tableau `notes`.

Cette opération est réalisée par la ligne :

```
notes[i] = sc.nextDouble();
```

Ensuite, je mets à jour la variable `somme` en y ajoutant la note saisie, ce qui me permet de calculer progressivement la somme totale des notes.

Cette addition est effectuée grâce à :

```
somme += notes[i];
```

La boucle est ensuite fermée avec `}`.

Une fois toutes les notes saisies, je calcule la moyenne en divisant la somme par 4, le nombre total de notes.

L'instruction utilisée est :

```
moyenne = somme / 4;
```

Enfin, j'affiche la moyenne calculée avec un message clair et compréhensible pour l'utilisateur.

Le code correspondant est :

```
System.out.println("La moyenne est : " + moyenne);
```

Pour terminer, j'ai fermé le scanner avec la méthode `close()` afin de libérer les ressources associées à la lecture clavier.

La dernière instruction est :

```
sc.close();
```

---

## Conversion script BP2->Question3 JAVA

```
Algorithme age()
Variables
    age : entier
Début
    Ecrire "Entrez votre âge"
    Lire age
    Si age < 18 Alors
        Afficher "Vous êtes mineur"
    Sinon Si age > 40 Alors
        Afficher "Vous êtes vétérân"
    Sinon
        Afficher "Vous êtes senior"
    Fin Si
Fin
```

```
import java.util.Scanner;

public class Age {
    public static void main(String[] args) {
        int age;
        Scanner scanner = new Scanner(System.in);

        System.out.print("Entrez votre âge : ");
        age = scanner.nextInt();

        if (age < 18) {
            System.out.println("Vous êtes mineur.");
        } else if (age > 40) {
            System.out.println("Vous êtes vétérân.");
        } else {
            System.out.println("Vous êtes senior.");
        }

        scanner.close();
    }
}
```





Pour transformer cet algorithme en code Java, j'ai commencé par déclarer une variable entière appelée `age` pour stocker l'âge que l'utilisateur va saisir, comme dans l'algorithme. J'ai ensuite utilisé un objet `Scanner` pour lire cette saisie depuis la console, ce qui correspond à l'instruction `lire age`. Avant cela, j'ai affiché un message avec `System.out.println("Entrez votre age")` pour inviter

l'utilisateur à entrer son âge, comme dans l'algorithme où on écrit `ecrire "entrez votre age"`. Pour vérifier les différentes conditions selon l'âge, j'ai utilisé une structure `if...else if...else` qui suit exactement la logique de l'algorithme : si l'âge est inférieur à 18, on affiche "Vous êtes mineur", si l'âge est supérieur à 40, on affiche "Vous êtes vétérân", sinon on affiche "Vous êtes senior". Enfin, j'ai fermé le scanner avec `scanner.close()` pour bien terminer la lecture des données. Ainsi, chaque étape de l'algorithme a été traduite simplement en Java tout en respectant la logique initiale.

# Réponse au Quizz du TP

1. Le langage machine le plus basique niveau circuit est  
**a. Le langage machine**
  2. Les langages qui permettent d'utiliser un vocabulaire qui utilise les termes :  
read, write ou add sont :  
**b. Haut niveau**
  3. Les règles du langage de programmation constituent :  
**a. la syntaxe**
  4. Un \_\_\_\_\_ traduit les instructions de langage de haut niveau en code  
machine  
**c. un compilateur**
  5. Les emplacements de mémoire nommés de l'ordinateur sont appelés :  
**b. variables**
  6. Les opérations individuelles utilisées dans un programme informatique sont  
souvent regroupées en unités logiques appelées :  
**a. procédures**
  7. Une instance de classe est :  
**c. un objet**
  8. Java a une architecture  
**a. neutre**
  9. Vous devez compiler les classes écrites en Java dans  
**a. un bytecode**
  10. Toutes les instructions de programmation Java doivent se terminer par :  
**c. un point-virgule**
-

## Exercice 1. Identifiants de classes

Identifiant	Statut
maClasse	Légal mais non conventionnel
void	 Illégal (mot réservé)
Golden Retriever	 Illégal (espace interdit)
invoice#	 Illégal (# interdit)
36535CodePos tal	 Illégal (commence par un chiffre)
Appartement	 Légal et conventionnel
Fruit	 Légal et conventionnel
8888	 Illégal (que des chiffres)
EcranTotal()	 Illégal (parenthèses interdites)
Acompte_rece vable	Légal mais non conventionnel



## Exercice 2. Identifiants de méthodes

Identifiant	Statut
<code>associationRoles()</code>	 Légal et conventionnel
<code>void()</code>	 Illégal (mot réservé)
<code>Golden Retriever()</code>	 Illégal (espace interdit)
<code>invoice#()</code>	 Illégal (# interdit)
<code>24500CodePostal()</code>	 Illégal (commence par un chiffre)
<code>PayrollApp()</code>	Légal mais non conventionnel
<code>getReady()</code>	 Légal et conventionnel
<code>911()</code>	 Illégal (commence par chiffre)
<code>EcranTotal()</code>	Légal mais non conventionnel

## Exercice 3.ParoleChanson.java

Pour cet exercice, j'ai écrit une classe Java nommée **ParoleChanson** qui affiche quatre lignes de paroles de la chanson « **La trace** » de l'artiste TK.

Description du programme

- La classe contient une méthode `main`, qui est le point d'entrée de tout programme Java.
- Dans cette méthode, j'utilise la commande `System.out.println()` pour afficher du texte dans la console.
- Chaque appel à `println()` affiche une ligne différente des paroles de la chanson.
- Le programme affiche donc quatre lignes de texte, chacune correspondant à une phrase tirée de la chanson.
- Ce programme simple permet de montrer comment afficher du texte en Java en utilisant des instructions `println()`.

```
public class ParoleChanson {  
    public static void main(String[] args) {  
        System.out.println("J'lui ai froissé sa carte bleue ce gros KV");  
        System.out.println("Il était à découvert mais il était en LV");  
        System.out.println("J'te prête plus X-ADV, chaque fois j'reçois des PV");  
        System.out.println("Déjà qu'à la base tu devais, on va finir par te crever");  
    }  
}
```

- `public class ParoleChanson` : déclaration de la classe appelée `ParoleChanson`.
- `public static void main(String[] args)` : déclaration de la méthode principale qui est exécutée au démarrage du programme.
- `System.out.println()` : méthode qui affiche un texte à l'écran, suivi d'un retour à la ligne.
- Chaque ligne de la chanson est affichée par une instruction `println()` différente.

## Exercice 4.Modèle illustré

- Pour cet exercice, j'ai écrit une classe Java nommée TableEtChaises qui affiche une représentation d'une table entourée de chaises à l'aide de caractères X.
- Description du programme  
La classe contient une méthode `main`, qui est le point d'entrée de tout programme Java.
- Dans cette méthode, j'utilise la commande `System.out.println()` pour afficher du texte dans la console.
- Chaque appel à `println()` affiche une ligne différente correspondant à une chaise, la table, ou les chaises disposées devant la table.
- Le programme affiche donc plusieurs lignes de texte qui, ensemble, représentent une table avec des chaises autour.
- Ce programme simple permet de montrer comment afficher du texte en Java et comment organiser plusieurs lignes pour créer une représentation graphique simple avec des caractères.

```
1 public class TableEtChaises {
2     public static void main(String[] args) {
3         // Chaise gauche
4         System.out.println("X      X");
5         System.out.println("X      X");
6
7         // Table
8         System.out.println("XXXXXXXXXX");
9
10        // Chaise droite
11        System.out.println("X      X");
12        System.out.println("X      X");
13
14        // Chaises avant
15        System.out.println(" X  X  X");
16        System.out.println(" X  X  X");
17        System.out.println(" X  X  X");
18    }
19 }
20
```

## Exercice 5.Modèle illustré

- La classe contient une méthode `main`, qui est le point d'entrée de tout programme Java.  
Dans cette méthode, j'utilise des boucles `for` et la commande `System.out.print()` pour afficher les caractères dans la console.  
Une première boucle permet d'afficher les espaces avant les "T" pour centrer le triangle. Une deuxième boucle affiche les lettres "T" sur chaque ligne. Chaque ligne se termine par un saut de ligne avec `System.out.println()`.
- Le programme affiche donc plusieurs lignes de texte qui, ensemble, forment un triangle croissant de 7 lignes, la première ligne contenant un seul "T", et la dernière ligne sept "T".
- Ce programme simple permet de montrer comment afficher du texte en Java et comment organiser plusieurs lignes avec des boucles pour créer une représentation graphique simple avec des caractères.

```
1 // triangle.java
2 public class triangle {
3     public static void main(String[] args) {
4         int lignes = 7; // nombre de lignes du triangle
5
6         for (int i = 1; i <= lignes; i++) {
7             // Afficher les espaces avant les T pour centrer le triangle
8             for (int j = 1; j <= lignes - i; j++) {
9                 System.out.print(" ");
10            }
11            // Afficher les T
12            for (int k = 1; k <= i; k++) {
13                System.out.print("T");
14            }
15            // Passer à la ligne suivante
16            System.out.println();
17        }
18    }
19 }
```

## Exercice 6:

### Debug 1 JAVA

```
1 public class Debug1
2
3     /* This program displays a greeting */
4     public static void main(String[] args)
5     {
6         Systemoutprintln("Salut").
7     }
```

J'ai regardé le programme et j'ai vu plusieurs erreurs. Pour que ça marche, j'ai fait ça :

- J'ai mis des **accolades {}** **autour de la classe**, parce que c'est obligatoire en Java.
- J'ai corrigé `Systemoutprintln` en `System.out.println`, c'est comme ça qu'on écrit pour afficher du texte.
- J'ai ajouté un **guillemet à la fin de "Salut"** pour fermer la phrase.
- J'ai mis un **point-virgule ; à la fin de la ligne**, sinon le programme ne comprend pas.
- Et j'ai mis le commentaire au bon endroit, au-dessus de la méthode.

Après ça, le programme marche bien et affiche « Salut ».

```
1 public class Debug1
2 {
3     /* This program displays a greeting */
4     public static void main(String[] args)
5     {
6         System.out.println("Salut");
7     }
8 }
9
```

## Debug 2 JAVA

```
1 public class Debug2
2 {
3     /* This program displays some output
4     public static void main(String args)
5     {
6         System.out.println("Programmer en java est fun.");
7         System.out.println("Faire un programme");
8         System.out.println("peut être un challenge,");
9         System.out.prnitln("mais quand la syntaxe est correcte,");
10        System.out.println("c'est satisfaisant");
11    }
12 }
```

J'ai regardé le programme et j'ai vu plusieurs erreurs. Pour que ça marche, j'ai fait ça :

- J'ai fermé le commentaire avec `*/`, sinon il prend tout le reste du code.
- J'ai corrigé `String args` en `String[] args`, c'est comme ça qu'on écrit la méthode `main` en Java.
- J'ai corrigé une faute dans `System.out.prnitln`, ça s'écrit `System.out.println` pour afficher du texte.
- Le reste du code était bon, donc j'ai gardé les autres lignes comme elles étaient.

Après ça, le programme marche bien et affiche les phrases comme prévu.

```
1 public class Debug2
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Programmer en java est fun.");
6         System.out.println("Faire un programme");
7         System.out.println("peut être un challenge,");
8         System.out.println("mais quand la syntaxe est correcte,");
9         System.out.println("c'est satisfaisant");
10    }
11 }
12
```

## Debug 3 JAVA

```
1 public class Debug33
2 {
3     public static void main(String[] args)
4     {
5         System.Out.println("Derrière la rivière");
6         system.out.println("et au dela du bois");
7         SysTem.0ut.println("à la maison du garde nous irons");
8     }
9 }
```

J'ai regardé le programme et j'ai vu plusieurs erreurs. Pour que ça marche, j'ai fait ça :

- J'ai corrigé `System.Out.println`, parce que "Out" doit s'écrire avec un "o" minuscule, donc j'ai mis `System.out.println`.
- J'ai aussi corrigé `system.out.println`, parce que "System" doit commencer par une majuscule.
- Ensuite, j'ai corrigé `SysTem.0ut.println`, parce que "0ut" était écrit avec un zéro au lieu de la lettre "o", donc j'ai mis `System.out.println`.
- J'ai aussi ajouté un accent dans "au dela", pour que ça s'écrive correctement "au-delà", mais ça n'empêche pas le programme de fonctionner.

Le reste du code était bon, donc j'ai gardé les autres lignes comme elles étaient.

Après ça, le programme marche bien et affiche les phrases comme prévu.

```
1 public class Debug33
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Derrière la rivière");
6         System.out.println("et au-delà du bois");
7         System.out.println("à la maison du garde nous irons");
8     }
9 }
10
```

# Debug 4 JAVA

```
1 import javax.swing.JOptionPane;
2 public class Debug4
3 {
4     public static main(String[] args)
5     {
6         JOptionPane.showMessageDialog(null, 1er GUI program)!
7     }
8 }
```

J'ai regardé le programme et j'ai vu plusieurs erreurs. Pour que ça marche, j'ai fait ça :

- J'ai corrigé la ligne `public static main(String[] args)`, parce qu'il manquait le mot-clé `void`. J'ai donc mis `public static void main(String[] args)`.
- Ensuite, j'ai corrigé le texte affiché dans `JOptionPane.showMessageDialog`, parce que le message n'était pas entre guillemets, ce qui provoquait une erreur. J'ai donc mis `"1er GUI program"` entre guillemets.
- Il y avait aussi un point d'exclamation `!` à la fin de la ligne au lieu d'un point-virgule, donc je l'ai remplacé par `;`.
- Le reste du code était correct, donc je n'ai rien changé d'autre.

Après ça, j'ai compilé le programme, et maintenant il s'exécute bien : une fenêtre s'ouvre avec le message "1er GUI program".

```
1 import javax.swing.JOptionPane;
2
3 public class Debug4
4 {
5     public static void main(String[] args)
6     {
7         JOptionPane.showMessageDialog(null, "1er GUI program");
8     }
9 }
```



# Cas pratique

j'ai d'abord créé un projet Java.

J'ai fait une classe **Jeu.java** où j'ai utilisé `JOptionPane`.

Le programme commence par une petite boîte de dialogue qui dit de penser à un nombre entre 1 et 10.

Ensuite, j'ai généré un nombre aléatoire avec `Math.random()` et je l'ai affiché dans une deuxième boîte, comme ça on peut comparer.

Après, je suis passé à la partie **Yummy**.

J'ai créé un package qui s'appelle *Yummy*.

Dedans, j'ai fait trois fichiers :

- **Yummy.java** qui affiche juste la devise « *Yummy prépare les meilleurs plats pour vos fêtes* ».
- **Yummy2.java** qui affiche la même phrase mais avec une bordure en étoiles.
- **YummyFun.java** qui affiche la devise d'été « *Yummy, c'est fun au soleil* », avec une bordure en S.

Au final, ce TP m'a appris à afficher des dialogues en Java, à utiliser l'aléatoire et aussi à mieux organiser mon code dans un package.